NASA Contractor Report 182086

ICASE INTERIM REPORT 12

LANZ: SOFTWARE SOLVING THE LARGE SPARSE SYMMETRIC GENERALIZED EIGENPROBLEM

Mark T. Jones
Merrell L. Patrick

NASA Contract No. NAS1-18605
August 1990

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center, Hampton, Virginia 23665

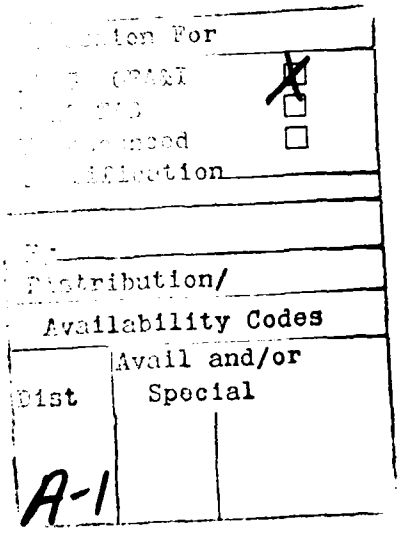
Operated by the Universities Space Research Association

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

ICASE INTERIM REPORTS

ICASE has introduced a new report series to be called ICASE Interim Reports. The series will complement the more familiar blue ICASE reports that have been distributed for many years. The blue reports are intended as preprints of research that has been submitted for publication in either refereed journals or conference proceedings. In general, the green Interim Report will not be submitted for publication, at least not in its printed form. It will be used for research that has reached a certain level of maturity but needs additional refinement, for technical reviews or position statements, for bibliographies, and for computer software. The Interim Reports will receive the same distribution as the ICASE Reports. They will be available upon request in the future, and they may be referenced in other publications.



Robert G. Voigt
Director

LANZ: Software for Solving the Large Sparse Symmetric Generalized Eigenproblem[1]

Mark T. Jones
Argonne National Laboratory
Argonne, IL 60439-4844
and
Merrell L. Patrick
Duke University
Durham, NC 27706

ABSTRACT

A package, LANZ, for solving the large symmetric generalized eigenproblem is described. The package has been tested on four different architectures: Convex 200, CRAY Y-MP, Sun-3, and Sun-4. The package uses a version of Lanczos' method and is based on recent research into solving the generalized eigenproblem.

[1]This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-18605 and the Air Force Office of Scientific Research under AFOSR grant No. 88-0117 while the authors were in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23665. Additional support was provided by NASA grant No. NAG-1-466. Computer time on a CRAY Y-MP was provided by the North Carolina Supercomputing Center.

1 Purpose

LANZ solves the symmetric generalized eigenproblem,

$$Kx = \lambda Mx, \tag{1}$$

where K is symmetric positive definite and M is positive semi-definite. It is also capable of solving

$$Kx = -\lambda Mx, \tag{2}$$

where M can be indefinite. It can find either 1) all the eigenpairs in a user-specified range, or 2) the p eigenpairs closest to some user-specified value, σ.

2 Method

LANZ is an implementation of the algorithm described in [5]. The heart of LANZ is an implementation of the Lanczos algorithm used with a spectral transformation similar to that described in [7]. LANZ uses an improved version of the selective orthogonalization algorithm proposed in [8] to maintain semi-orthogonality among the Lanczos vectors.[1] In addition, LANZ uses a dynamic shifting algorithm to accelerate convergence to desired eigenpairs in a slightly different fashion than in [2].

To utilize the spectral transformation, LANZ requires the solution of linear systems of the form $(K - \sigma M)$x=b, where the matrix $(K - \sigma M)$ can be indefinite.[2] When LANZ can determine that $(K - \sigma M)$ is positive definite, an algorithm designed for banded positive definite linear systems on supercomputers is used. If LANZ determines that the matrix may be indefinite, then an algorithm for solving banded indefinite linear systems on supercomputers is used [6][4]. It is assumed that the sparse matrices have been reordered to have a small bandwidth via an available reordering algorithm. Alternatively, the linear system solution algorithms included with LANZ can be replaced with ones of the user's choice, a relatively simple task that is described in documentation accompanying the code.[3]

[1] The improved version of the selective orthogonalization algorithm is described in [3].

[2] When solving Equation 2, the system to be solved is $(K + \sigma M)x = y$.

[3] The authors expect to add a sparse factorization and solution algorithm for indefinite linear systems to LANZ in the near future.

3 Description of the Package

LANZ has been coded in FORTRAN with some well-documented machine-dependent routines for memory allocation and timing written in C. LANZ has been tested on four different computers: the CRAY Y-MP running UNICOS, the Convex 220 running CONVEX UNIX, the Sun-3, and the Sun-4 running SunOs UNIX. LANZ was originally designed to run on computers that have vector instructions such as the CRAY Y-MP and, as such, is optimized for vector machines. In the CRAY Y-MP and Convex 200 versions of LANZ, integers are 8 bytes long; in the Sun-3 and Sun-4 versions, integers are 4 bytes long. Reals are 8 bytes long in all versions of the code.

3.1 Installation

Because the code must include different compiler directives for the CRAY Y-MP and Convex 220, as well as other different features for the different machines, a software development utility, called MAX, is used and included with the package [1]. The installation process is completely automated by the use of makefiles and scripts and is described in documentation accompanying the source code.

4 Usage

To use the LANZ package, the user can either 1) link to the compiled library, lanz.a, and directly call the LANZ subroutine as described in the following section, or 2) run the driver provided with the program and read in matrices from files.[4] Because the driver program is fairly long and is described in documentation accompanying the source code, it is not included in this paper.

4.1 Call to LANZ

CALL LANZ(ipar,rpar,theta,bj,y,relate,K,Krp,Kcp,M,Mrp,Mcp,guess)

The parameters ipar, rpar, K, Krp, Kcp, M, Mrp, and Mcp must be set by the user on input. The parameters theta, bj, y, relate, and guess are optionally set by the user on input. Upon return, the parameters ipar, rpar, theta, bj, y, relate, and guess contain information generated by LANZ.

[4] The use of the driver program is described in documentation accompanying the code.

4.2 Description of Parameters

ipar	in/out	ipar is an integer array of dimension 19 used to set many of the options for LANZ. Upon return, it contains information about the execution of LANZ.
ipar(1)	in	The order of the matrices, K and M.
ipar(2)	in	The maximum number of eigenpairs that can be stored. This is limited by the size of y, relate, theta, and bj. This parameter should be set to more than the number of eigenvalues one is seeking (or expects to find).
ipar(3)	in	The number of eigenvalues being sought. This value is ignored if ipar(12) is 1.
ipar(4)	in/out	On input this is the maximum number of Lanczos steps that LANZ can take. On output it is set to the number of steps LANZ took. This value only determines the amount of time one is willing to wait for an answer; a suggested default is three times the number of desired eigenpairs.
ipar(5)	out	0 if LANZ was successful, a negative value otherwise.
ipar(6)	in/out	On input it is set to the number of eigenpairs already found (and stored in theta, bj, y, and relate). On output it contains the total number of eigenpairs found (including those already present prior to execution of LANZ).
ipar(7)	in	Sets the level of debugging output desired. This should normally be set to 0 (positive values result in more debugging output, none of which is of interest to the user).
ipar(8)	in	The type of problem being solved: 0) K is positive definite and M is positive semi-definite ($Kx = \lambda Mx$), 1) K is positive definite, M is indefinite, and the shift σ is non-zero ($Kx = -\lambda Mx$), or 2) same as 1) except that σ is zero.

ipar(9) in — A parameter that instructs LANZ to use inertia calculations to ensure that the eigenvalues that were found were actually those closest to the desired σ. A 0 indicates no inertia checking, and a 1 indicates that inertia checking is desired. If LANZ determines that some eigenvalues are missing, it will automatically attempt to find them. A word of caution: normally this option requires an extra factorization of $(K-\sigma M)$, an expensive calculation. Also, this option is redundant if ipar(12) is set to 1.

ipar(10) in — Indicates the type of printed report that LANZ should produce. For no output, it should be set to 0. Otherwise, the type of report produced is determined by summing the numbers of the following options: 2) print the orthogonality matrix of the computed eigenvectors (Y^TY), 4) print the number of negative eigenvalues to the left of each shift point used, 16) print the eigenvalues, the estimated and calculated error bounds on each eigenpair[a], 32) print the eigenvalues and their estimated error bounds, 64) print only the eigenvalues. For example, if the report is to contain the number of negative eigenvalues to the left of each shift (option 4) and the eigenvalues (option 64), then this parameter would be set to 68.

[a] The calculated error bound on an eigenpair is found by $\| Ky-\lambda My \| / | \lambda |$.

ipar(11) in — The maximum number of steps to take on one shift. This value is important only in that it is used by LANZ to determine how much memory it needs to allocate. A suggested default is 50. This value can be increased if LANZ suggests it, or decreased if LANZ indicates that it has run out of memory.

ipar(12) in — This parameter indicates whether all the eigenvalues in a range are sought (set to 1), or if a fixed number of eigenvalues nearest to a given shift are sought (set to 0).

ipar(13) in — Storage format for K. Should always be set to 0 in this version of LANZ.

ipar(14)	in	Storage format for M. Should always be set to 0 in this version of LANZ.
ipar(15)	in	The level of loop unrolling to be used in the factorization and triangular matrix solution steps for positive definite matrices. Legal values are 1, 4, and 6. The suggested value is 6.
ipar(16)	in	The type of factorization used. A 0 indicates that LANZ should decide between a LDL^T decomposition and a Bunch-Kaufman type algorithm for indefinite matrices. A 1 indicates that in the interests of speed that a LDL^T decomposition is to always be used. It is suggested that this value be left at 0.
ipar(17)	in	A value of 0 leaves dynamic shifting on, a value of 1 turns it off. It is suggested that this value be left at 0 unless the user understands what the dynamic shifting algorithm does and decides that it is undesirable for his/her purpose.
ipar(18)	in	A value of 0 indicates that no initial guess for an eigenvector exists. A value of 1 indicates that an initial guess for an eigenvector is stored in the parameter, guess.
ipar(19)	in	This value should be set to the leading index of the y array.
rpar	input	rpar is an real array of dimension 5 that is used to set some of the options for LANZ.
rpar(1)	in	The σ or shift to be searched around, ignored if ipar(12) is 1.
rpar(2)	in	The desired relative accuracy of the eigenpairs, the formula for relative accuracy is given as a footnote to ipar(10). A suggested default is 1.
rpar(3)	in	If ipar(12) is set to 1, this value is the left end of the range in which to search.
rpar(4)	in	If ipar(12) is set to 1, this value is the right end of the range in which to search.

rpar(5) in The value is the storage factor to be used when allocating space for the factored matrix when performing a Bunch-Kaufman factorization. This value is multiplied by the amount of space required for a positive definite factorization of $(K - \sigma M)$ to get the amount of space to be allocated for a Bunch-Kaufman type factorization. A suggested default is 1.1 and should be changed only if LANZ suggests it.

theta in/out Theta is an array of real values of at least length ipar(2). On input, it is used to store any eigenvalues found prior to calling LANZ. On output it is used to store any eigenvalues found by LANZ. The eigenvalues stored prior to a call to LANZ will not be written over (this also pertains to bj, y, and relate).

bj in/out bj is an array of real values of at least length ipar(2). On input, it is used to store the error bounds on any eigenvalues found prior to calling LANZ. On output it is used to store the error bounds on any eigenvalues found by LANZ. Entry i in bj is the error bound for the eigenvalue in entry i in theta.

y in/out y is a two-dimensional array of real values with ipar(19) rows and at least ipar(2) columns. On input, the columns of y are used to store any eigenvectors found prior to calling LANZ. On output the columns of y are used to store any eigenvectors found by LANZ. Eigenvectors are related to the corresponding eigenvalues by the relate parameter.

relate in/out Relate is an integer array of length at least ipar(2) that describes the relationship of eigenvector to its corresponding eigenvalue. Eigenvalue i corresponds to eigenvector relate(i). On input relate stores the relationships of eigenpairs found prior to the call to LANZ, and on output it stores the relationships of pairs found by LANZ.

K	in	K stores the nonzero entries in the K matrix. The first ipar(1) values should be the diagonal entries of K (including zeros). The next number of non-zero entries entries should be the off-diagonal non-zeroes in the upper triangle of K stored by row. The amount of memory allocated for K should be 2*(ipar(1)+the number of non-zero off-diagonals in the upper triangle of K). This extra memory is used as working space and is a tradeoff to increase execution speed on vector computers. K and all the information associated with it (as well as M) are left intact on output.
Krp	in	Krp is an integer array that indicates where each row of non-zeroes begins in K (with an assumed offset of ipar(1)). For example, if entry *i* in Krp is set to 10, it indicates that the non-zeroes for row *i* start at position ipar(1)+10 in K. The amount of memory allocated for Krp should be 2*(ipar(1)+1).
Kcp	in	Kcp is an integer array that indicates the column number of each non-zero off-diagonal in K. For example, a value of 10 in Kcp(30) indicates that the non-zero in position K(ipar(1)+30) is in column number 10. The amount of memory allocated for Kcp should be ipar(1)+2*(number of off diagonal non-zeroes in the upper triangle of K).
M	in	M stores the nonzero entries in the M matrix. The first ipar(1) values should be the diagonal entries of M (including zeros). The next number-of-non-zero-entries entries should be the off-diagonal non-zeroes in the upper triangle of M stored by row. The amount of memory allocated for M should be 2*(ipar(1)+the number of non zero off diagonals in the upper triangle of M). This extra memory is used as working space and is a tradeoff to increase execution speed on vector computers. M and all the information associated with it (as well as M) are left intact on output.

Mrp	in	Mrp is an integer array that indicates where each row of non-zeroes begins in M (with an assumed offset of ipar(1)). For example, if entry i in Mrp is set to 10, it indicates that the non-zeroes for row i start at position ipar(1)+10 in M. The amount of memory allocated for Mrp should be 2*(ipar(1)+1).
Mcp	in	Mcp is an integer array that indicates the column number of each non-zero off-diagonal in M. For example, a value of 10 in Mcp(30) indicates that the non-zero in position M(ipar(1)+30) is in column number 10. The amount of memory allocated for Mcp should be ipar(1)+2*(number of off diagonal non-zeroes in the upper triangle of M).
guess	in/out	Guess is a real array of at least length ipar(1) that is used to store an initial guess for an eigenvector, if any. It will be destroyed on output.

Acknowledgment

The authors thank Eugene Poole for the use of his fast factorization/solution subroutines for banded linear systems.

References

[1] C. A. Felippa, *Utilities for Master Source Code Distribution: MAX and Friends*, Contractor Report 178383, CSM Branch, NASA Langley Research Center, Hampton, VA, 1988.

[2] R. G. Grimes, J. G. Lewis, and H. D. Simon, *The Implementation of a Block Lanczos Algorithm with Reorthogonalization Methods*, ETA-TR-91, Boeing Computer Servies, Seattle, WA, May, 1988.

[3] M. T. Jones, *The Use of Lanczos' Method to Solve the Generalized Eigenproblem*, PhD thesis, Department of Computer Science, Duke University, 1990.

[4] M. T. JONES AND M. L. PATRICK, *Bunch-Kaufman Factorization for Real Symmetric Indefinite Banded Matrices*, Technical Report 89-37, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA, 1989.

[5] ——, *The Use of Lanczos's Method to Solve the Large Generalized Symmetric Definite Eigenvalue Problem*, Technical Report 89-67, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA, 1989.

[6] ——, *Factoring Symmetric Indefinite Matrices on High-Performance Architectures*, Technical Report 90-8, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA, 1990.

[7] B. NOUR-OMID, B. N. PARLETT, T. ERICSSON, AND P. S. JENSEN, *How to Implement the Spectral Transformation*, Mathematics of Computation, 48 (1987), pp. 663–673.

[8] B. N. PARLETT AND D. S. SCOTT, *Lanczos Algorithm with Selective Orthogonalization*, Mathematics of Computation, 33 (1979), pp. 217–238.

NASA

Report Documentation Page

1 Report No.	2 Government Accession No.	3. Recipient's Catalog No.
NASA CR-182086 ICASE Interim Report 12		

4 Title and Subtitle	5. Report Date
LANZ: SOFTWARE FOR SOLVING THE LARGE SPARSE SYMMETRIC GENERALIZED EIGENPROBLEM	August 1990
	6. Performing Organization Code

7 Author(s)	8. Performing Organization Report No.
Mark T. Jones Merrell L. Patrick	Interim No. 12
	10. Work Unit No.
	505-90-21-01

9 Performing Organization Name and Address	11. Contract or Grant No.
Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225	NAS1-18605
	13. Type of Report and Period Covered
12 Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	Contractor Report
	14 Sponsoring Agency Code

15 Supplementary Notes

Langley Technical Monitor:
Richard W. Barnwell

Interim Report

16 Abstract

A package, LANZ, for solving the large symmetric generalized eigenproblem is described. The package has been tested on four different architectures: Convex 200, CRAY Y-MP, Sun-3, and Sun-4. The package uses a version of Lanczos' method and is based on recent research into solving the generalized eigenproblem.

17 Key Words (Suggested by Author(s))	18. Distribution Statement
generalized Eigenvalue problem, Lanczos method	39 - Structural Mechanics 61 - Computer Programming and Software Unclassified - Unlimited

19 Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	13	A03